Vue.js – Handout

Pascal Göbel, Daniel Becker, Simon Kiefer

Vue.js ist ein progressives JavaScript-Framework für **Single Page Applications (SPAs)**.
Kernprinzipien:

- **Single File Components (SFCs)**: Alles (HTML, JS, CSS) in einer `.vue`-Datei.
- **Kapselung**: Jede Komponente ist autark (Struktur, Logik, Design).
- **Nachrüstbar**: Von kleiner Bibliothek bis zu vollständiger SPA skalierbar.

Vorteile:

- Leichtgewichtig.
- Reaktives Rendering ohne Boilerplate.
- Exzellente Performance durch Virtual DOM.

Grundgerüst einer `.vue`-Datei:

```
<template>
  <!-- HTML mit Vue-Directives -->
</template>

<script setup>
  // JavaScript-Logik
</script>

<style scoped>
```

```
  /* Scoped CSS */
</style>
```

- `<template>`: Deklaratives HTML mit Mustache-Syntax `{{ variable }}`.
- `<script setup>`: Moderne Composition API (empfohlen seit Vue 3).
- `<style scoped>`: CSS gilt nur für diese Komponente.

Basis der Reaktivität: `ref()` aus Vue importieren.

Beispiel:

```
<script setup>
import { ref } from 'vue'
const myVar = ref('World')  // .value im Script
</script>

<template>
  <div>Hello {{ myVar }}!</div>  <!-- Direkter Zugriff im Template -->
</template>

<style scoped>
div { color: red; }
</style>
```

- **Script**: `myVar.value` manipulieren.
- **Template**: Automatisch reaktiv mit `{{ myVar }}`

Props leiten Daten von **Eltern** zu **Kindern** weiter.

Elternkomponente (Sender)

```
<template>
  <MyButton :count="42" :user="{ name: 'Max' }" />
</template>
```

- Kurzform: `:prop="wert"` (v-bind).

Kindkomponente (Empfänger)

```
<script setup>
const props = defineProps(['count', 'user'])
console.log(props.user.name)  // "Max"
</script>

<template>
  <div>Count: {{ count }}</div>
</template>
```

Emits senden Events **von Kind zu Eltern**.

Kindkomponente

```
<script setup>
const emit = defineEmits(['click'])
const handleClick = () => emit('click', { id: 1 })
</script>

<template>
```

```
  <button @click="handleClick">Klick mich!</button>
</template>
```

Elternkomponente

```
<template>
  <Child @click="onChildClick" />
</template>

<script setup>
const onChildClick = (data) => console.log(data.id)  // 1
</script>
```

`$event`: Übergebene Daten empfangen.

Wichtige Vue-Directives für DOM-Manipulation:

Directive	Verwendung	Beispiel
`v-if` / `v-else`	Bedingtes Rendering (DOM entfernt)	`<div v-if="ok">Ja</div>`
`v-for`	Listen	`<div v-for="n in [1,2,3]">{{ n }}</div>`
`v-model`	2-Wege-Binding (Inputs)	`<input v-model="text">`
`v-bind` (`:`)	Props binden	`<img :src="imageUrl">`
`v-on` (`@`)	Events	`<button @click="doIt">Klick</button>`
`v-show`	Toggle per CSS	`<div v-show="ok">Immer im DOM</div>`

`v-model` intern: `:value + @update:modelValu`

Erweiterte Features

Auswahl relevanter Konzepte:

Computed & Watch

```
import { ref, computed, watch } from 'vue'
const count = ref(5)
const doubled = computed(() => count.value * 2)  // Automatisch
reaktiv
watch(count, (newVal) => console.log(newVal))    // Bei Änderung
```

Offizielle Docs:

- vuejs.org/guide/introduction
- vuejs.org/api
- pinia.vuejs.org/
- router.vuejs.org/